Results of December 2021 LunarG Vulkan Ecosystem & SDK Survey

Public Report

Karen Ghavam, LunarG
January 2022

Table of Contents

Executive Summary

This report provides the results from the LunarG SDK & Ecosystem Survey completed in December of 2021.

Methodology

LunarG and Khronos developer relations created this Vulkan ecosystem survey to gauge the Vulkan community's use of and satisfaction with the current Vulkan ecosystem. This is a follow-up survey to the previous LunarG survey completed in November 2021 (fifth annual Vulkan ecosystem survey).

We hoped to reach as many Vulkan developers as possible -- both SDK users and non-SDK users -- by promoting the survey on Twitter, Reddit, LinkedIn, the Khronos DevRel slack channel, and the Vulkan Discord server. As well it was sent directly to 13,000+ recipients of the LunarG LunarXchange Vulkan SDK mailing list.

Survey Highlights

1. There were 288 respondents (vs. 364 the previous year).
2. LunarG mailing list, Reddit, and Twitter are where the majority of respondents heard about the survey.
3. 53% of the respondents were self-study developers. 47% of the respondents were developers developing for commercial purposes.
4. Compared to the previous year survey (commercial developers):
    a. *Released applications* have risen from 37% to 43%.
    b. *Planning for future release* has risen from 23% to 38%.
5. Target platforms for applications are (in order of usage) Windows, Desktop Linux, Android, macOS, SteamDeck, and iOS.
6. It is time for LunarG to deprecate Visual Studio 2015 support for the SDK. Everybody was using VS2017 or later (except for about 5 respondents), and the majority of users were on the newest version of Visual Studio. Moving forward on supported versions of Visual Studio will help control maintenance costs.
7. Satisfaction in the completeness of Validation Layer coverage (high rating) has improved 15% (up from 48% to 63%).
8. Khronos resources that are used to learn Vulkan:
    a. Vulkan specification is by far the most used resource for learning Vulkan (80% of respondents).
    b. Khronos Samples (25%), Khronos Vulkan Guide (23%), Khronos Vulkan website (21%), Khronos webinars and videos (20%) are all in 2nd place.

    c. Khronos Vulkan slack channel, Vulkan discord, and the Khronos Vulkan Forum are unknown by a significant number of respondents.
9. Non-Khronos resources that are used to learn Vulkan:
    a. Sascha Willems' examples (48%) and vulkan-tutorial.com (50%) are the most popular.
    b. It should be noted that all of the Sascha Willems' examples are now in the KhronosGroup/Vulkan-Samples repository as well.
10. Vulkan Configurator: Compared to the previous year survey:
    a. Overall familiarity with vkconfig has increased from 47% to 59%.
    b. Favorable *Usefulness* score has increased from 49% to 60%.
11. GFXReconstruct usage compared to the previous year has increased from 9% of respondents to 24%.
12. Synchronization validation continues to be a "top hitter" in usage and satisfaction. And respondents want more…

Open-ended Feedback

There were a lot of open-ended comments received from the respondents. This section highlights open-ended feedback that was mentioned multiple times.

- How can the validation layers be improved?
    - Error Reporting:
        - There is a project in the Validation Layer repository tracking enhancements to the formatting of validation layer errors.
        - Some of the comments indicated that individuals were not aware of validation layer error reporting improvements and debug utils white paper. These resources could prove helpful to these individuals.
        - Multiple individuals struggle with interpreting errors from the validation layers. We have ideas for tools to help with this. However, it hasn't made our to-do list yet.
    - Performance
        - Validation layers performance needs improvement. We have been working on performance improvements over the last year and have a significant improvement recently merged (fine-grain locking).
        - But there is still more to do…
    - Synchronization
        - The validation of Vulkan synchronization has been very helpful to many individuals and this was stated multiple times.
        - Request for synchronization validation across command buffers in a queue. This is under development and will be released in the near future.

    - Better coverage
        - Although the survey feedback indicates that the validation layers are finding most of their application issues and rarely give false positives, we know there are many gaps to still be filled.
- What features, tools, and/or improvements would you like to see added to the LunarG Vulkan SDK (Windows, Linux, and/or macOS)?
    - Requests for more/better/improved tutorials and samples. The SDK no longer provides samples or tutorials because it is not considered a tool for teaching the Vulkan API, but rather a tool for developing to the Vulkan API. There are Khronos initiatives to improve samples (KhronosGroup/Vulkan-Samples) and tutorials
    - Requests for the Vulkan Memory Allocator to be added to the SDK. Unfortunately, resources did not allow for integration of the Vulkan Memory Allocator, Vulkan Hardware Capability Viewer, and VOLK into the SDK during 2021. This year's survey also reveals high interest in GLFW and SDL.
    - Requests for consistent tagging of SDK releases. A recent improvement in the SDK release process results in an SDK tag added to all Khronos repositories except for the KhronosGroup/SPIRV-Cross and the KhronosGroup/SPIRV-Reflect repositories. The branches are named SDK-X.Y.ZZZ (where X.Y.ZZZ is the Vulkan Header version for the SDK) and the tags are on that branch named SDK-X.Y.ZZZ.release where the release is usually “0” but will be incremented if additional SDKs are released for that header version.
    - Request for more complete macOS support using MoltenVK. During 2022 LunarG will be investing resources to enable GPU-AV and Debug Printf on macOS as well as to develop a loader for iOS.
    - Requests for Vulkan synchronization debugging tools (online or offline).
    - Request for a tool for inspecting calls from GFXReconstruct trace.
        - Note: A new tool under development, *toascii*, should provide this capability. When we are confident in its quality and reliability, it will be added to the Vulkan SDK.
- What inhibits you from effectively and efficiently developing Vulkan applications?
    - Again, the request for more/better/improved samples, tutorials, and documentation.
    - Learning the Vulkan API takes significant time and effort.
    - VK_ERROR_DEVICE_LOST is hard to debug.
    - Shader debugging is a challenge.
    - GPU driver bugs can be a challenge.
    - Layer ecosystem issues. Bad implicit layers in the ecosystem can cause hard to diagnose problems.
    - Glslang missing features. The pipeline to compile GLSL into spir-v modules needs some improvement.

- Which aspects/features of developing with the Vulkan API do you like?
    - API Definition is great
        - Low-level approach and exactness of the API
        - Vulkan is a GPU specification
        - The amount of low-level control I now have
        - High-quality specification (precise, clear, complete)
        - Performance that the Vulkan API enables
        - Multi-threading
        - Independence of shader language
    - Cross-platform
    - Validation Layers and Tooling
        - Good error validation
        - Renderdoc!
        - Debug Printf
        - Synchronization validation
- Is there any feedback you would like to share?
    - Please make the chunked specification the default.
    - When are we going to get a cross-vendor mesh shading extension?
    - An official open-source library on top of Vulkan that could simplify things to a level like Metal
    - VK_KHR_dynamic_rendering was a big step in the right direction.

Planned Actions

LunarG and the Vulkan Working Group will prioritize the following projects in 2022:

1. Deprecate Visual Studio 2015
    a. SDK and several repositories will no longer build, test, and support VS2015
2. Validation layers. Keep doing what we are doing.
    a. Continue focus on validation layer performance initiative.
    b. Continue focus on synchronization validation implementation.
    c. Continue GitHub issue responsiveness and improved coverage.
    d. Avoid releasing extensions without timely validation layer coverage.
3. Vulkan learning resources (tutorials, samples, documentation, …)
    a. We added 21 samples to the repo in 2021. For 2022 more complex samples will be explored.
    b. Explore gathering some high-quality, non-Vulkan-specific resources to aid developers new to graphics programming.
    c. Evaluate currently available best practice documentation and potential ways to improve its visibility.
    d. Explore collaborations with external tutorials such as www.vulkan-tutorial.com to improve the quality of the tutorial by adding new content and/or making sure existing content is accurate.

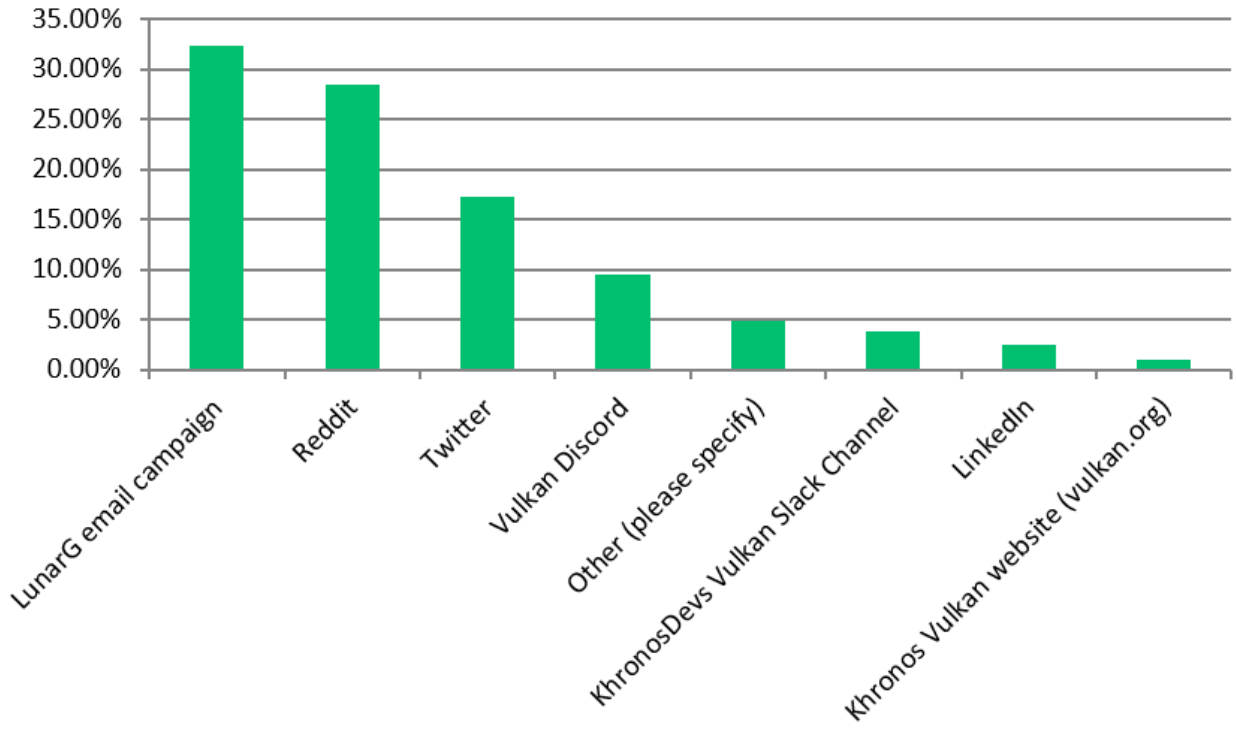
e. Continue Vulkanised webinars with a focus on educational content. All content is made available for free on the Vulkan Youtube channel.

4. During 2022, LunarG will prioritize the following initiatives:
    a. Addition of VOLK and VMA to the SDK
    b. Addition of GPU-AV and Debug Printf support to the macOS SDK
    c. Addition of support for iOS as a target (in other words, create a loader for iOS)

Survey Results: Survey Questions and Responses

The remainder of this document contains the December 2021 Vulkan Ecosystem and SDK Survey questions and corresponding responses. Where appropriate we have included respondents' written answers from the *Other* response category as well as our own clarifying comments.

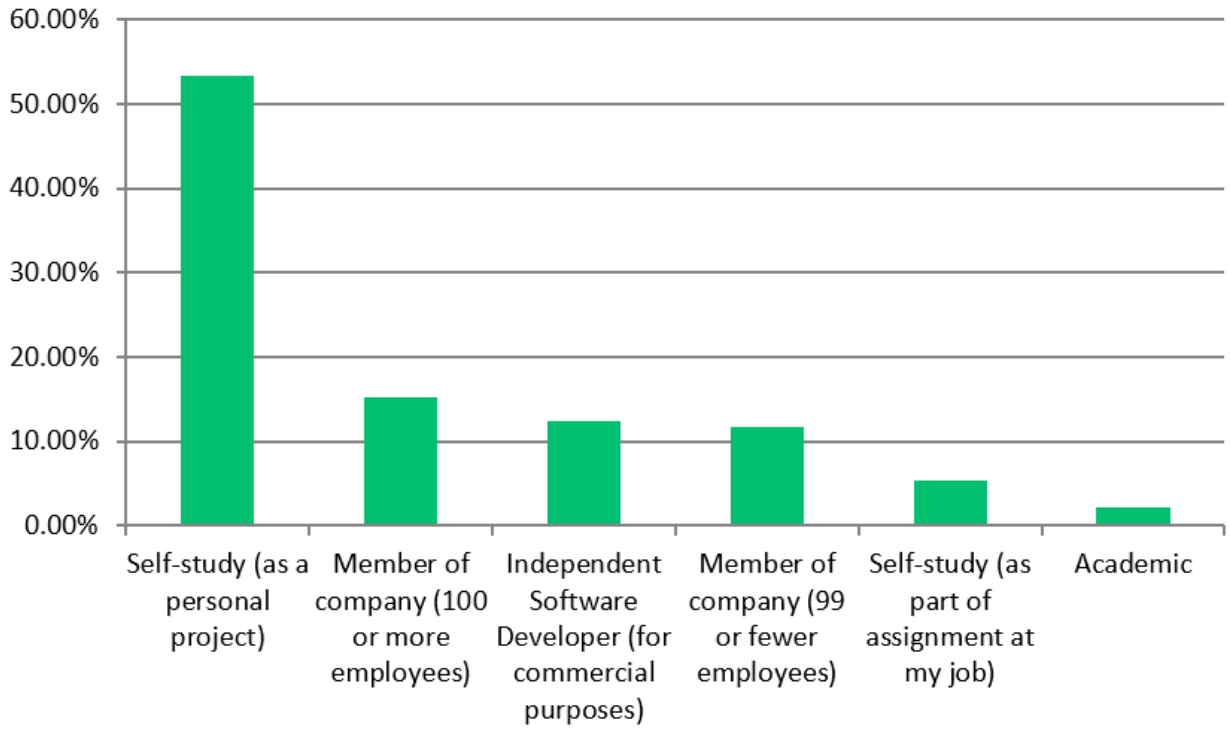
Where did you hear about this survey?



Responses in the *Other* category include:

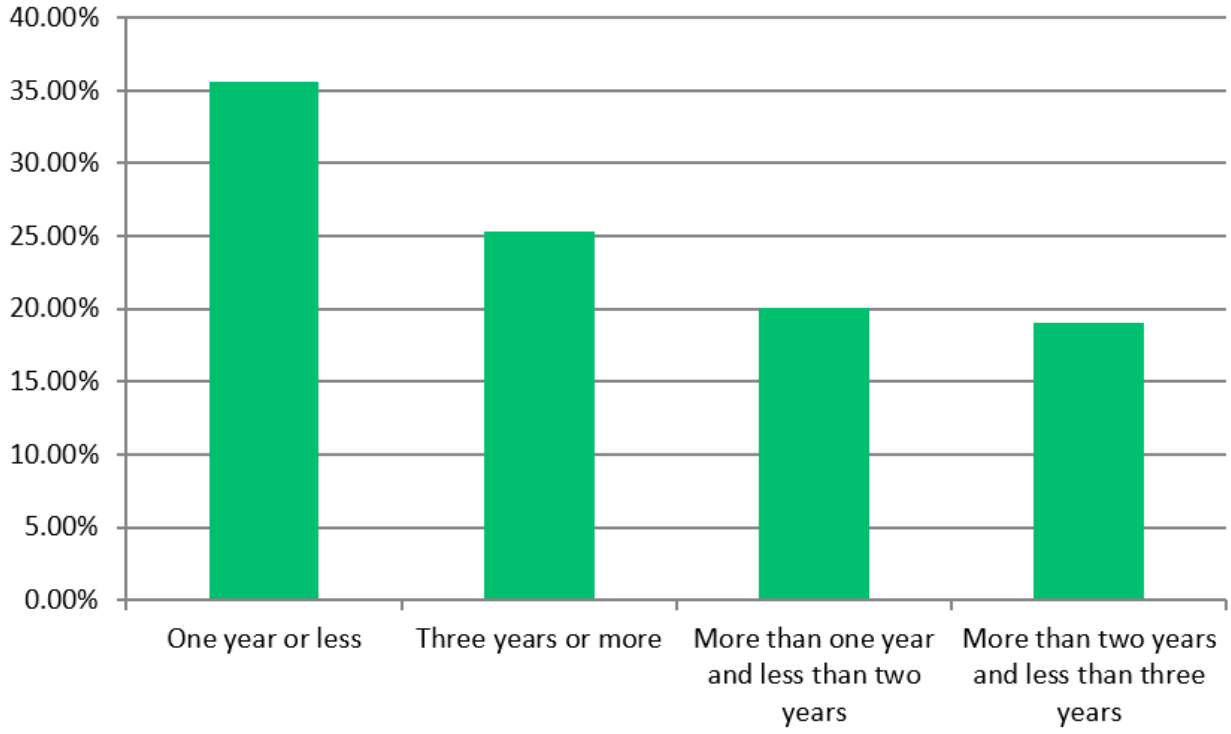
1. Gamedev.ru (5)
2. Khronos newsletter (2)
3. Telegram (4)
4. From work (1)
5. Friend (1)

What type of Vulkan developer are you?

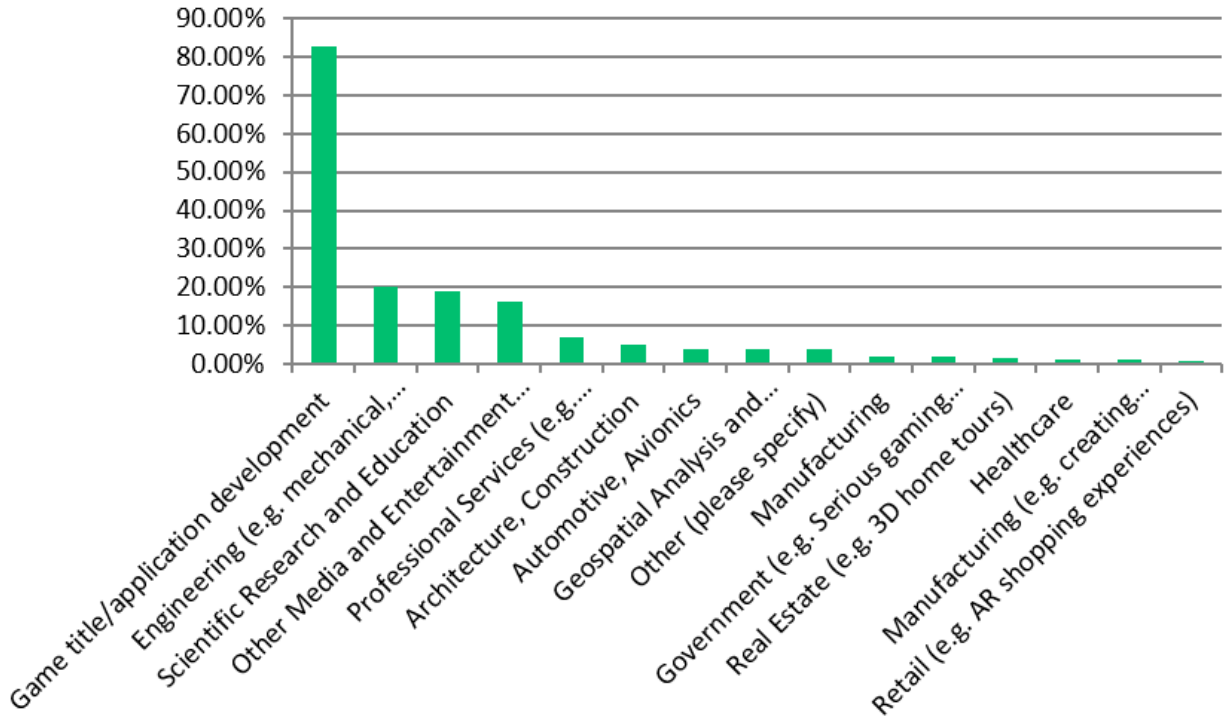


Note: Throughout this report, the results were evaluated for two populations: 1) all respondents and 2) developers working for commercial companies. In most cases, the commercial companies' population answers were not significantly different than the entire population. 53% of the respondents were self-study developers. 47% of the respondents were developers developing for commercial purposes.

How long have you been developing with the Vulkan API?

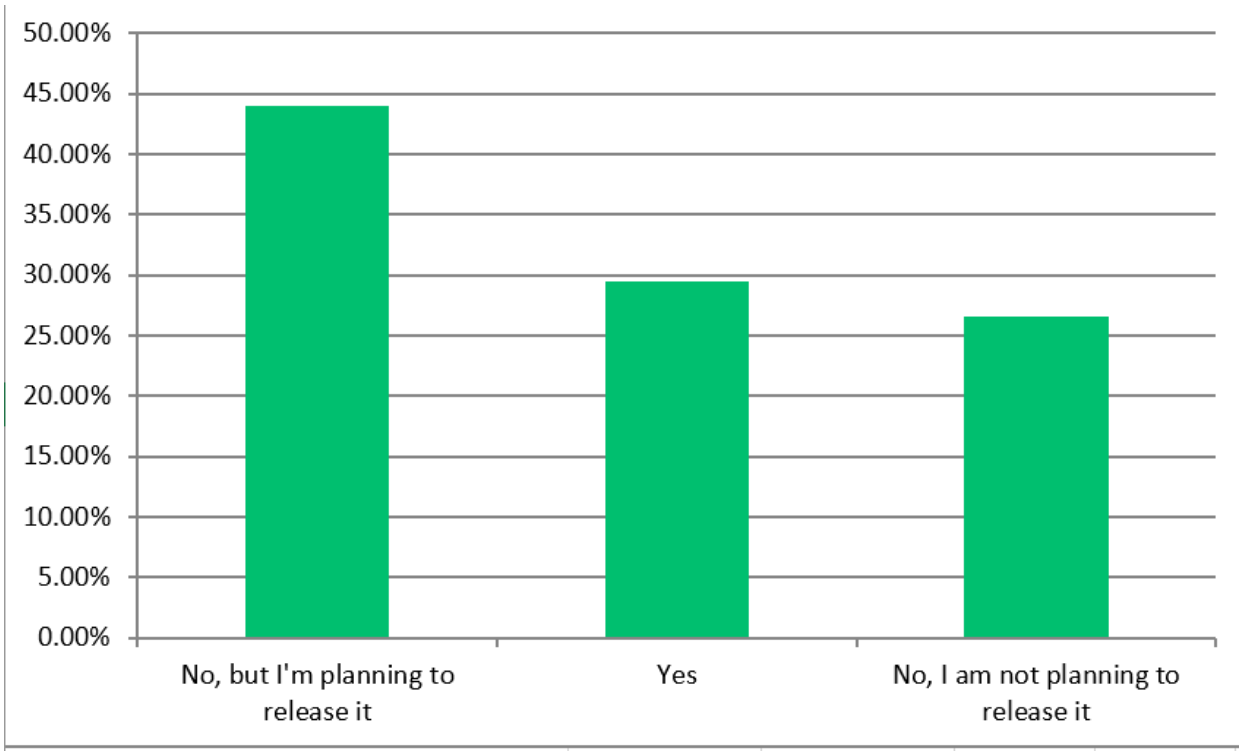


Your development is for what type of industry? Check all that apply:



Note: Although game title and game application development is the top industry for Vulkan development, over the last two years, Engineering and Scientific Research and Education have increased from 12% and 10% to 20% and 19% respectively.

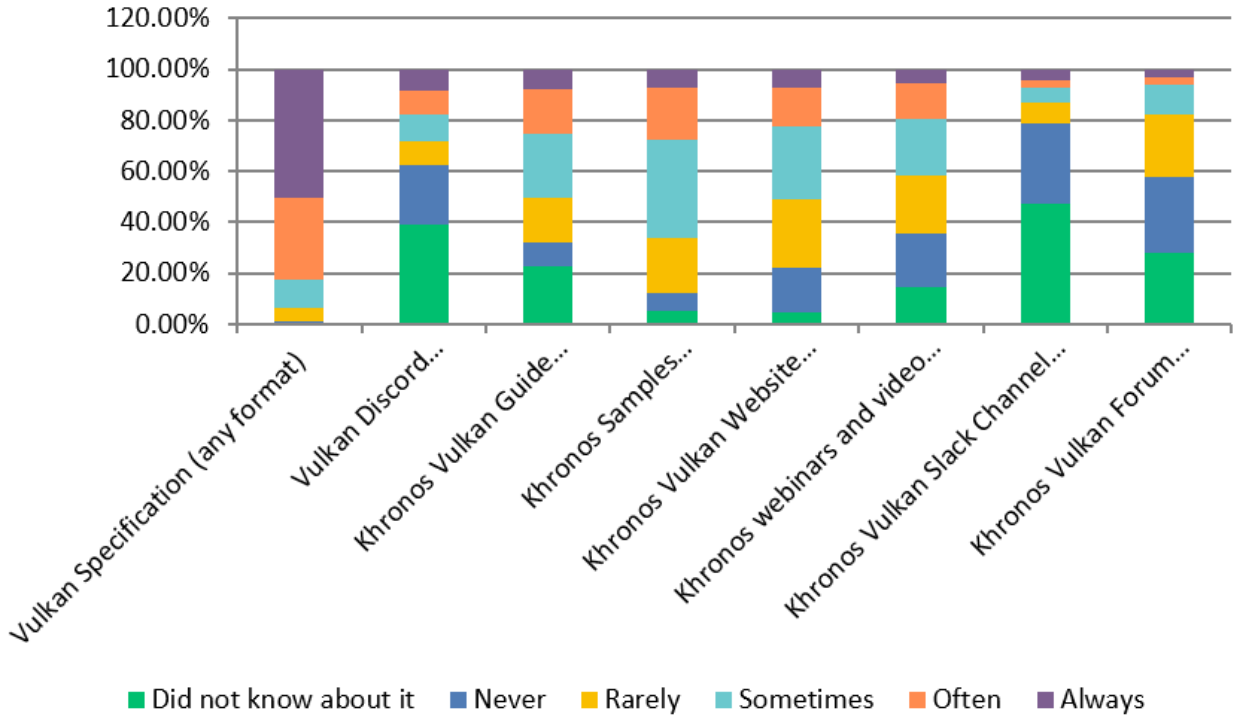
Have you released your Vulkan development project for public use?



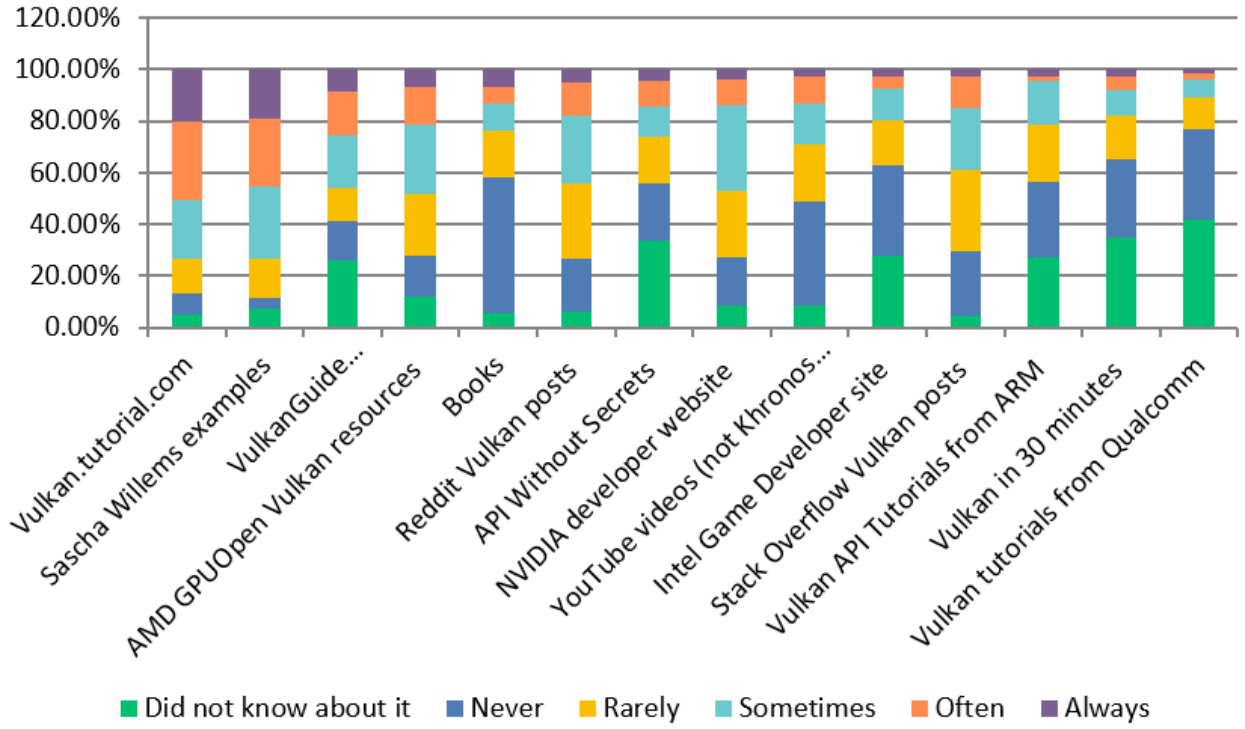
Notes:

1. Compared to the previous year survey (commercial developers):
    a. Those respondents indicating released applications have risen from 37% to 43%.
    b. Those respondents planning for future release have risen from 23% to 38%

Which Khronos resources did you use to learn the Vulkan API?

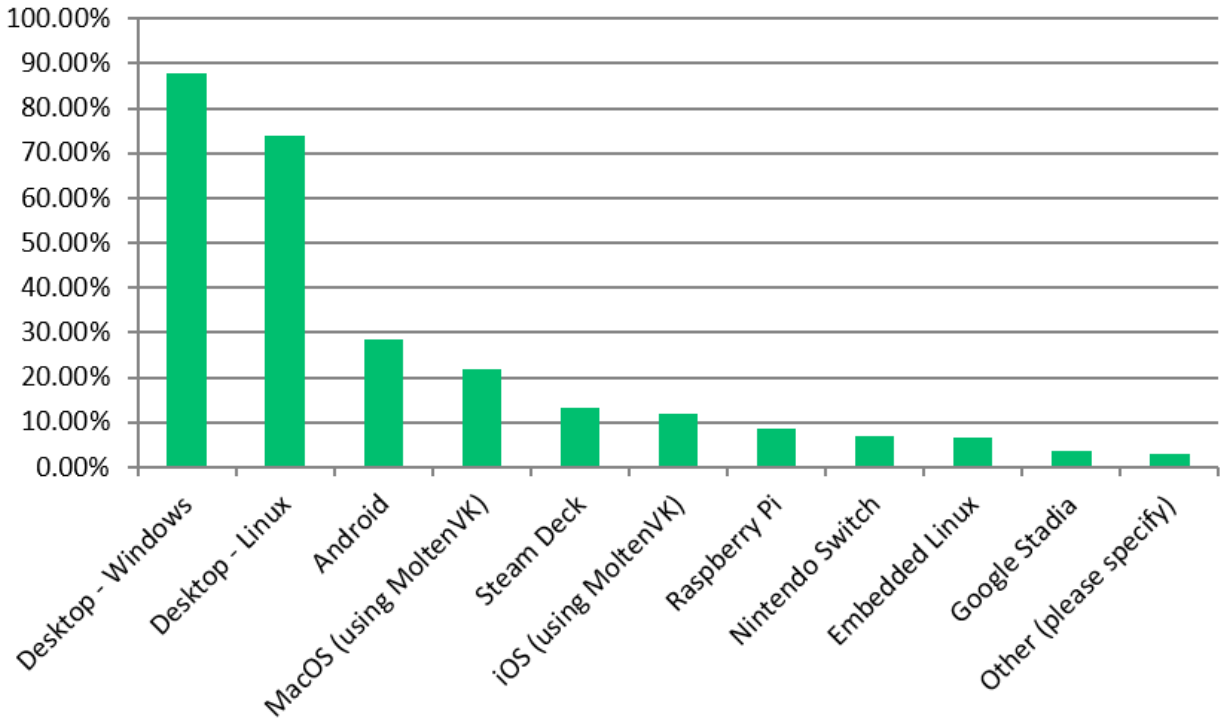


Which resources (non-Khronos) did you use to learn the Vulkan API?

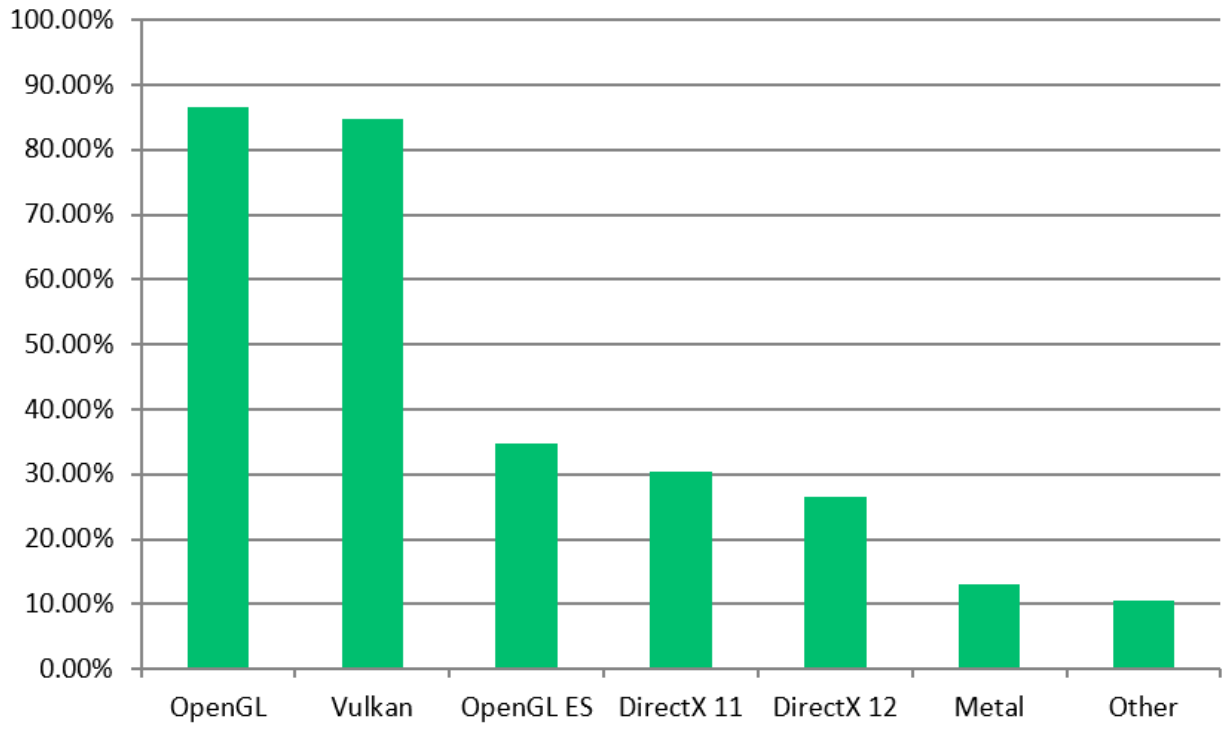


Note: All of the Sascha Willems examples have been migrated to the KhronosGroup/Vulkan-Samples repository. As well, Khronos has begun collaboration with vulkan.tutorial.com to help improve the tutorial.

What is the target of your Vulkan application? Check all that apply:



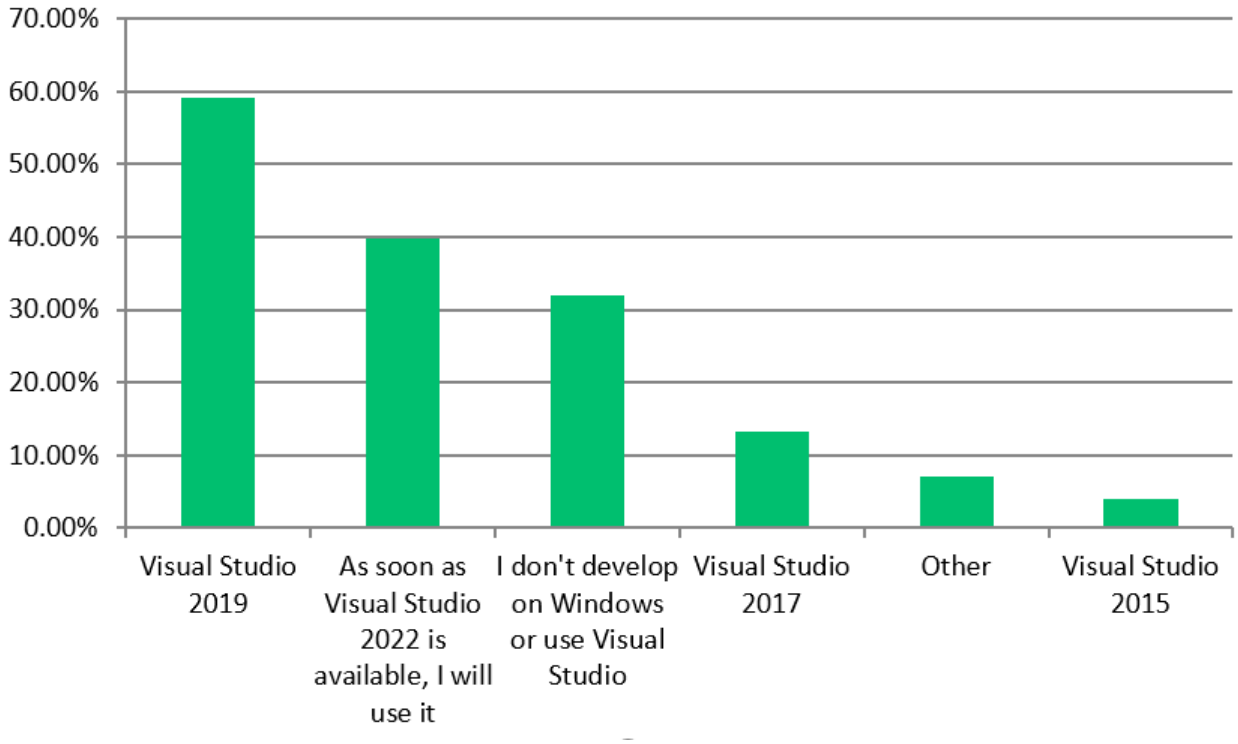
Which graphics API do you have experience with? Check all that apply:



Responses in the *Other* category include:

1. WebGPU
2. WebGL
3. Consoles
4. Direct3D 9, Direct3D10.

Which version of Visual Studio are you using? Check all that apply:

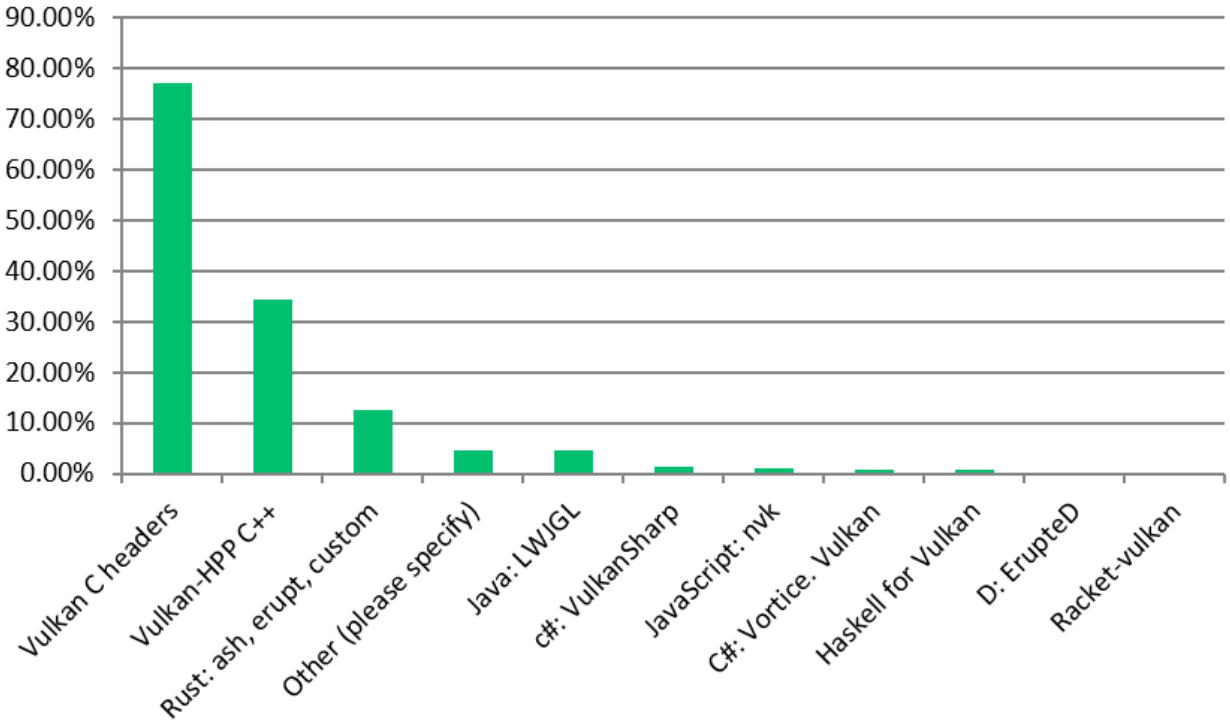


Responses in the *Other* category include:

1. Visual Studio code (most of them)
2. Visual Studio 2013 (one)

Note: We will begin the deprecation process for VS 2015 to help reduce our support/maintenance costs.

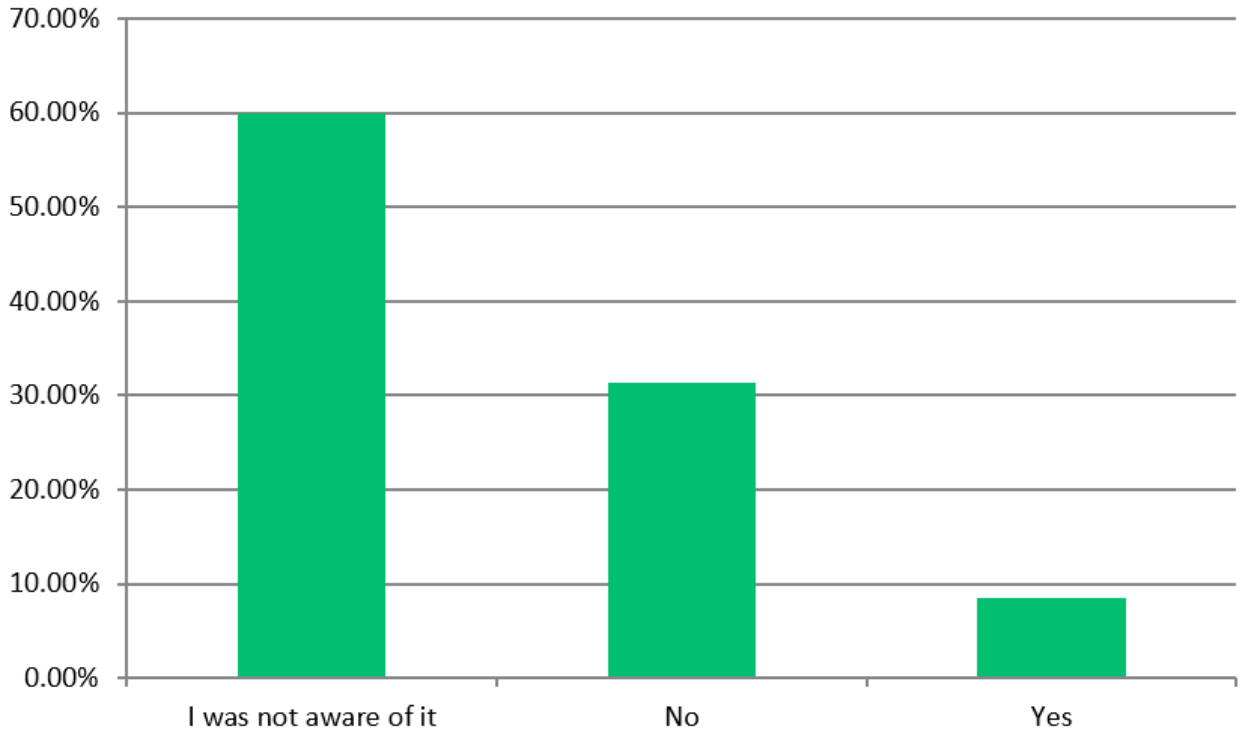
What language binding do you use? Check all that apply:



Responses in the *Other* category include:

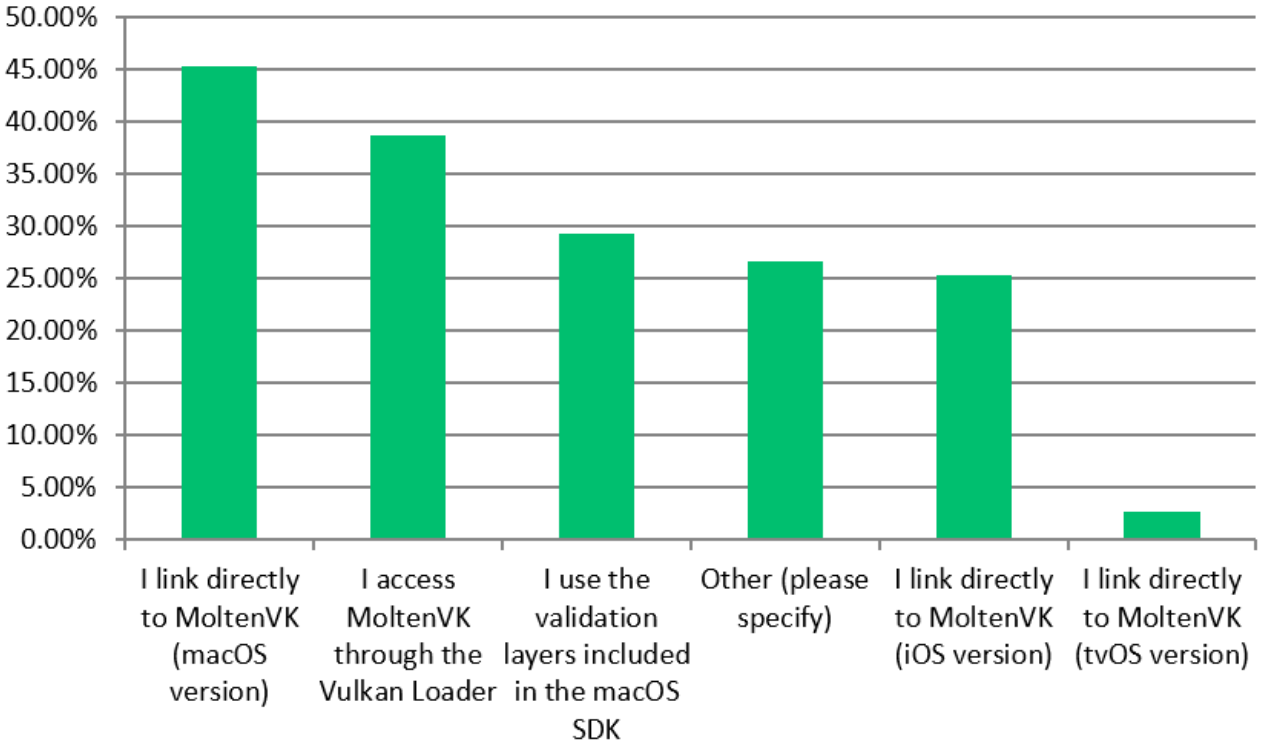
1. Custom C++ headers generated from vk.xml
2. C#: TerraFX (https://github.com/terrafx/terrafx.interop.vulkan)
3. Custom C# bindings

Do you use the *Unattended Install* feature of the Windows SDK installer?



Note: To learn more about the **unattended install option for the Windows SDK**, visit https://vulkan.lunarg.com/content/view/latest-sdk-version-api and the *Unattended Installation* section in the SDK Getting Started document.

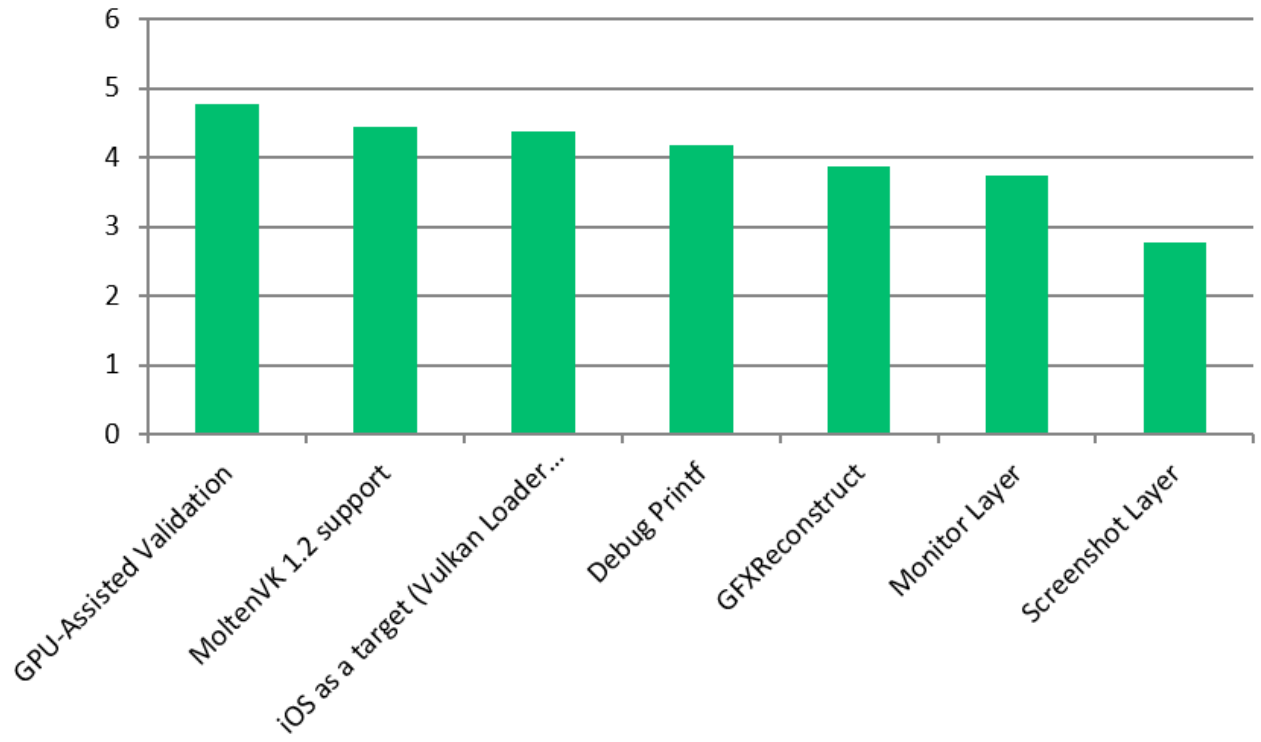
When doing Vulkan development for macOS, iOS, and/or tvOS, check all that apply.



Responses in the *Other* category include:

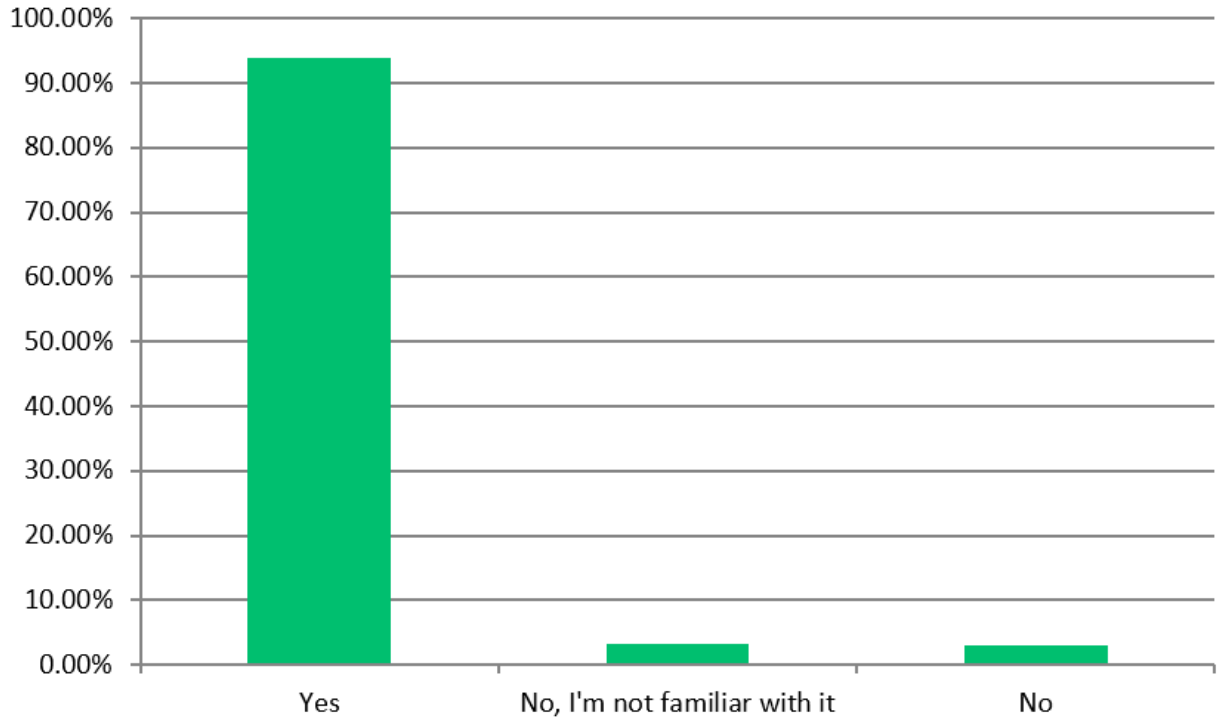
1. Multiple users indicated they use Metal directly on Apple platforms and not MoltenVK.
2. There were some users confused about how to use MoltenVK (link directly, use the loader?). *The State of Vulkan on Apple*, a whitepaper released by LunarG, provides information about developing using Vulkan in the Apple ecosystem.

Rank the importance of the following layers/tools in the macOS SDK?

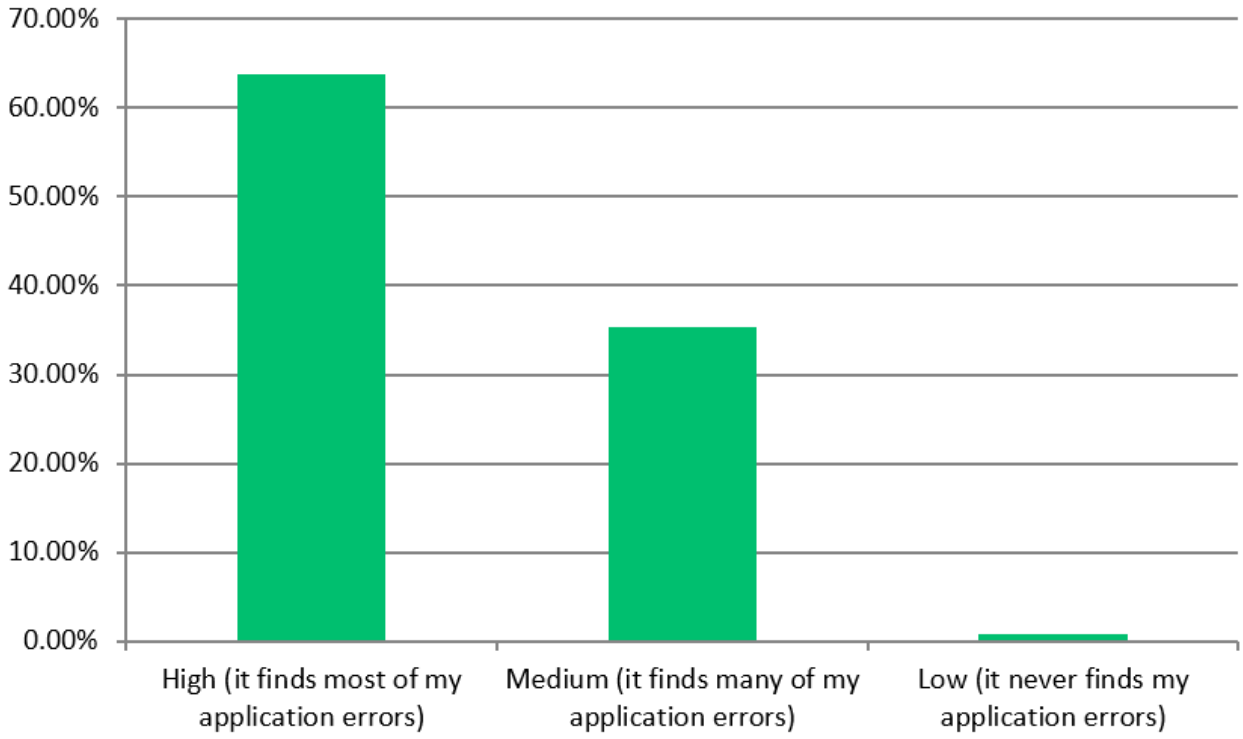


Note: In 2022, LunarG will begin efforts to add GPU-Assisted Validation and Debug Printf capabilities on macOS. In addition, we will create a loader for iOS.

Do you use the Khronos Vulkan Validation Layer (VK_LAYER_KHRONOS_validation)?

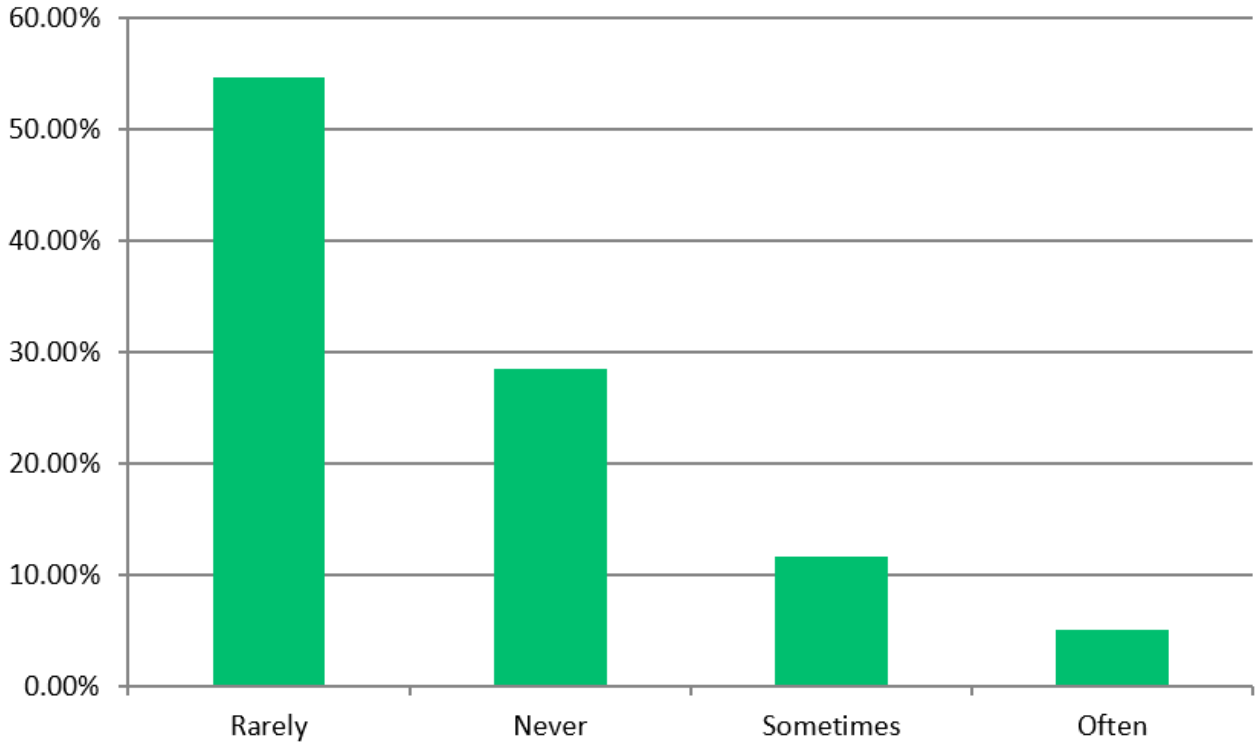


How would you rate the completeness of validation layer coverage?

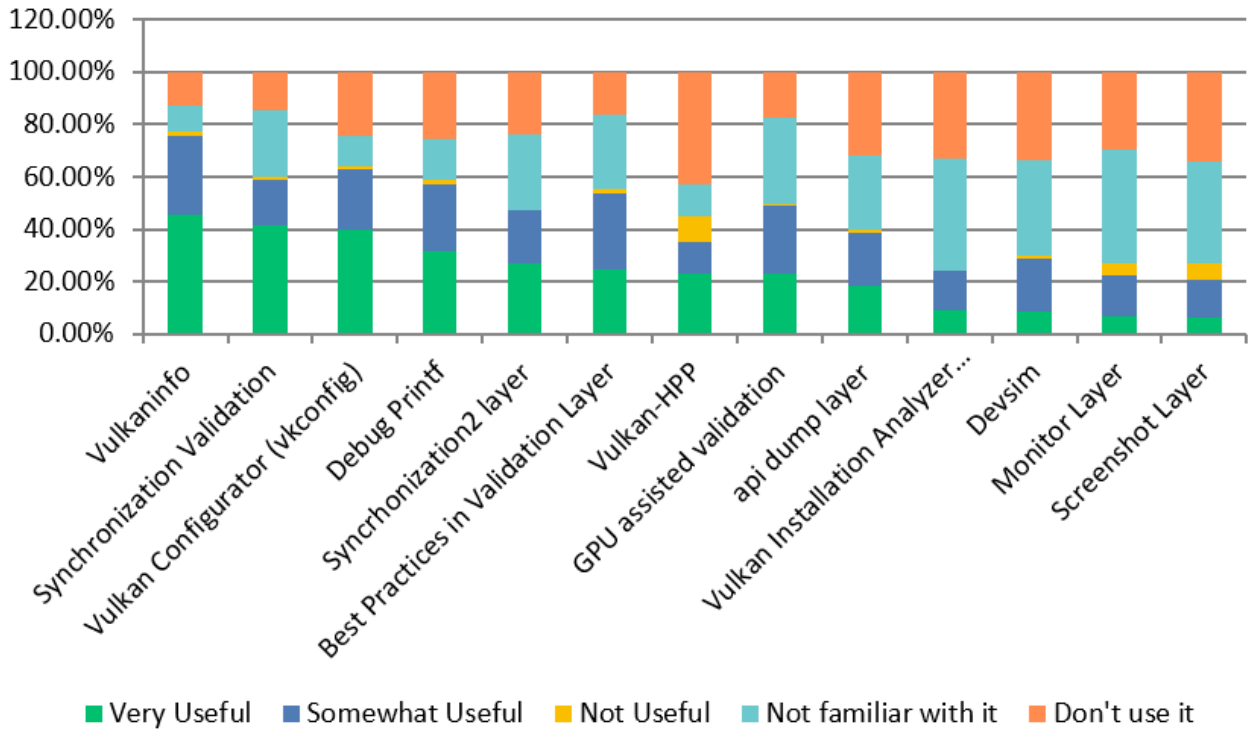


Note: Compared to last year's survey, the response *High* has moved up 15% from 48% to 63%

How often do you see false error reports (error reported when it shouldn't have been)?

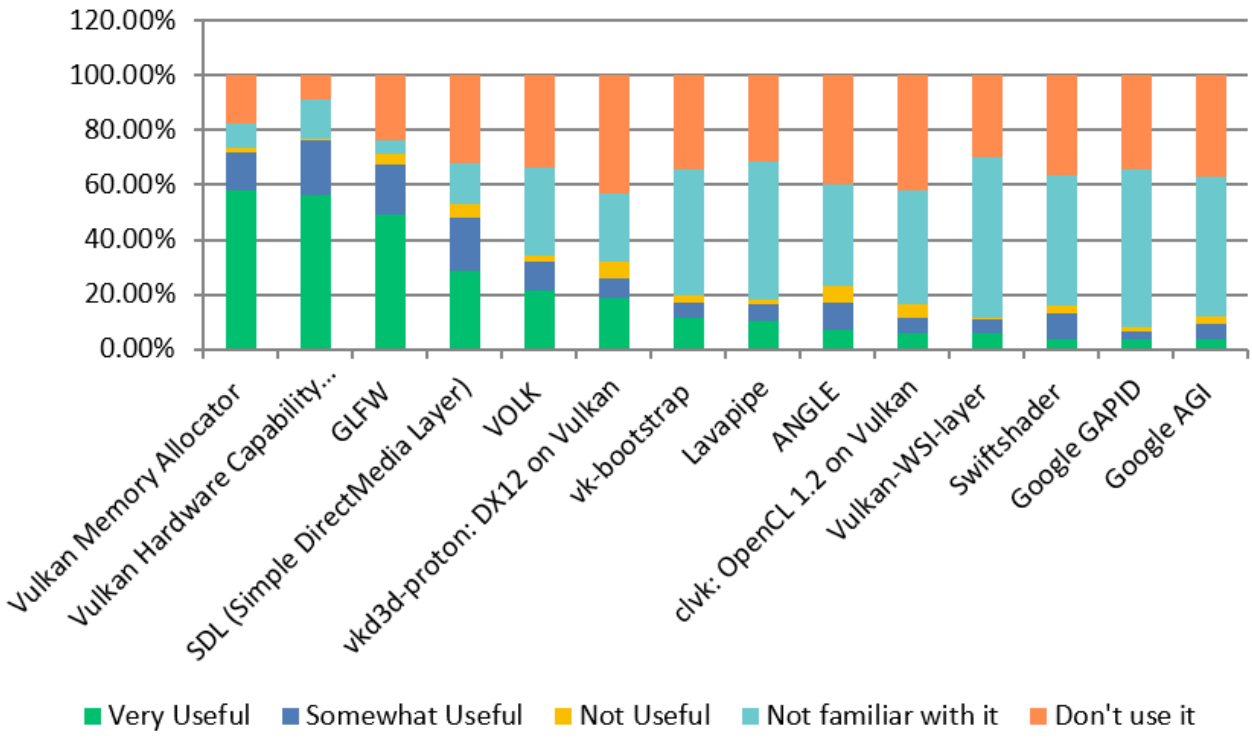


For the Vulkan layers, tools, or Validation Layer objects listed below, indicate their usefulness:



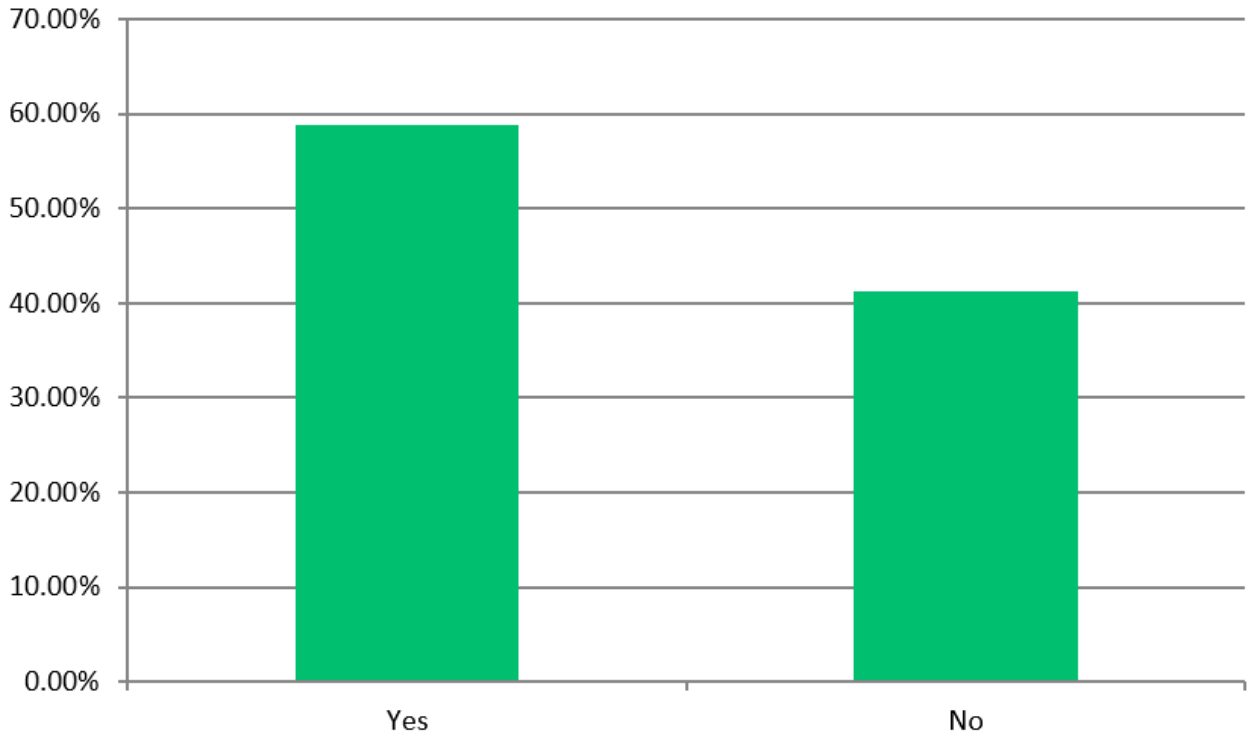
Note: Compared to last year's survey, more respondents are aware of the Vulkan Configurator (vkconfig). The ecosystem is discovering the usefulness of this tool.

For the Vulkan ecosystem tools/layers (not included in the SDK) that are listed below, indicate their usefulness:



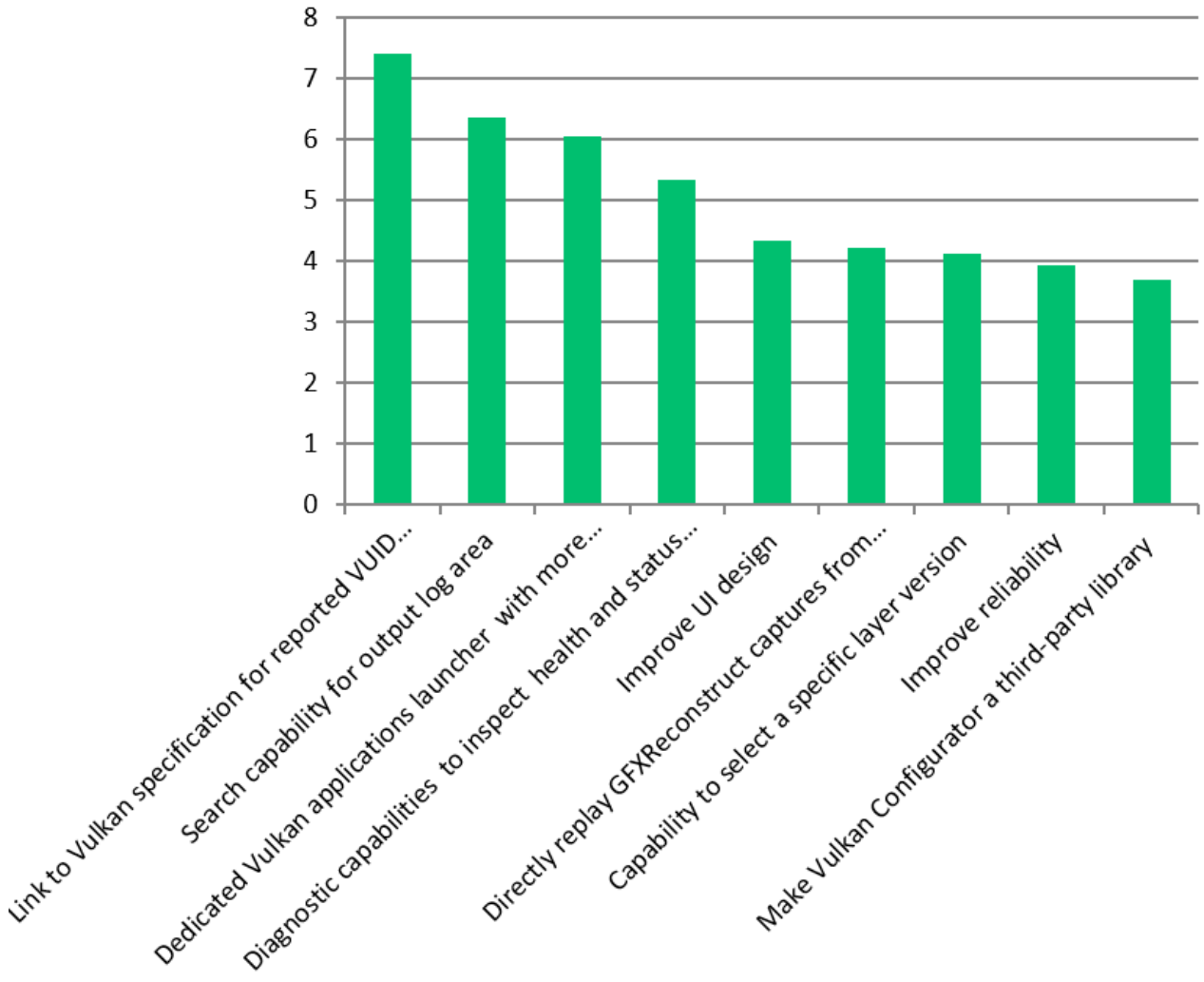
Note: Responses to this question are very similar to last year. Unfortunately, resources did not allow for integration of the Vulkan Memory Allocator, Vulkan Hardware Capability Viewer, and VOLK into the SDK. This year's survey also reveals high interest in GLFW and SDL.

Are you familiar with the Vulkan Configurator (vkconfig)?



Note: Compared to the last year's survey, the respondent's overall familiarity with the Vulkan Configurator (vkconfig) has increased from 47% to 59%. The ecosystem is discovering the usefulness of this tool.

Rank the following vkconfig improvement areas from 1 to 9 (1 is most important, 9 is least important)

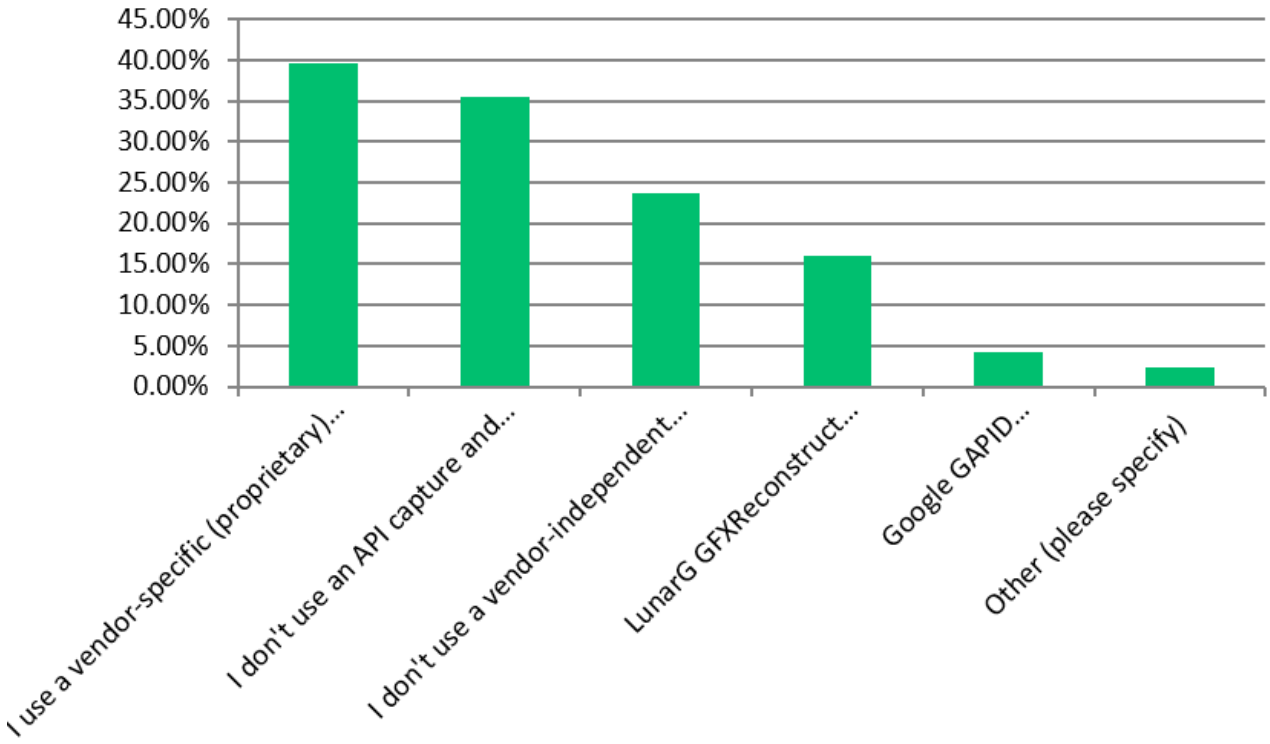


Do you have additional suggestions for the improvement of the Vulkan Configurator?

There were several comments about bad implicit layers having negative interactions with applications.

- Developers want to have the ability to disable these layers via vkconfig
    - This feature is available today
- Developers want to see the list of environment variables an application needs to set to opt-out of any implicit layers.
    - All implicit layer env variables are now logged by the loader (1.2.189 or later) by setting the loader debug environment variable *VK_LOADER_DEBUG=layer* or *VK_LOADER_DEBUG=all*. The loader also lists what layers are used during either CreateInstance or CreateDevice
- Developers want a “safe mode” loader that would automatically disable implicit layers. This was considered but due to bad unintended consequences for necessary and good implicit layers, it was decided not to implement this. See Vulkan-Loader issue #513.
    - Note: We are in the process of documenting the implicit layer issue in the ecosystem as well as solution alternatives that have been considered to increase ecosystem understanding of the situation and potential mitigations that can be taken.

Which vendor-independent tool do you use for multi-frame API capture and analysis? Check all that apply:



Note: Compared to the last year's survey, usage of GFXReconstruct has increased from 9% to 16% (all users) and 9% to 24% (commercial developers).

How would you rate the overall quality of the Vulkan ecosystem today?